Bing

I see that you want to implement an Angular TreeNode component based on the Json data structure you provided. This is a common task for web developers who want to display hierarchical data in a tree-like format.

To implement an Angular TreeNode component, you need to follow these steps:

- Create a TreeNode class that represents a single node in the tree. The class should have properties for id, label, icon, expanded, selected, actionButtons, and children. The children property should be an array of TreeNode objects.
- Create a TreeComponent that receives its model from outside using the @Input decorator. The model should be an array of TreeNode objects that represents the root nodes of the tree. The component should use the ngFor directive to iterate over the model and render each node using a custom template.
- Create a custom template for the TreeComponent that displays the label, icon, and action buttons of each node. The template should also use the ngIf directive to check if the node has any children and if it is expanded. If both conditions are true, the template should recursively render the children nodes using the TreeComponent itself.
- Add some logic to the TreeComponent to handle the events such as expanding, collapsing, selecting, and dragging the nodes. You can use the @Output decorator to emit custom events to the parent component or use the @ViewChild decorator to access the TreeModel instance and call its methods.

Here is an example of how the code might look like:

```
// tree-node.ts
export class TreeNode {
  id: number;
  label: string;
  icon: string;
  expanded: boolean;
  selected: boolean;
  actionButtons: any[];
  children: TreeNode[];

  constructor(options: any) {
    this.id = options.id;
    this.label = options.label;
    this.icon = options.icon;
    this.expanded = options.expanded;
    this.selected = options.selected;
    this.actionButtons = options.actionButtons;
    this.children = options.children.map((child: any) => new TreeNode(child));
  }
}

// tree.component.ts
import { Component, Input, Output, EventEmitter } from '@angular/core';
import { TreeNode } from './tree-node';

@Component({
  selector: 'app-tree',
  templateUrl: './tree.component.html',
  styleUrls: ['./tree.component.css']
})
export class TreeComponent {
  @Input() model: TreeNode[]; // the model received from outside
  @Output() nodeSelected = new EventEmitter<TreeNode>(); // a custom event to emit when a node is selected

  // a method to toggle the expanded state of a node
  toggleExpanded(node: TreeNode) {
    node.expanded = !node.expanded;
  }

  // a method to select a node and emit the nodeSelected event
  selectNode(node: TreeNode) {
    node.selected = true;
    this.nodeSelected.emit(node);
  }

  // a method to handle the drag start event of a node
  onDragStart(event: DragEvent, node: TreeNode) {
    event.dataTransfer.setData('text', node.id.toString()); // set the node id as the drag data
  }

  // a method to handle the drag over event of a node
  onDragOver(event: DragEvent, node: TreeNode) {
    event.preventDefault(); // prevent the default behavior of dropping
  }

  // a method to handle the drop event of a node
  onDrop(event: DragEvent, node: TreeNode) {
    event.preventDefault(); // prevent the default behavior of dropping
    const sourceId = event.dataTransfer.getData('text'); // get the source node id from the drag data
    const sourceNode = this.findNodeById(sourceId); // find the source node by id
    if (sourceNode && sourceNode !== node) { // check if the source node exists and is not the same as the target node
      this.moveNode(sourceNode, node); // move the source node to the target node
    }
  }

  // a helper method to find a node by id in the model
  findNodeById(id: string, nodes: TreeNode[] = this.model): TreeNode | null {
    for (const node of nodes) {
      if (node.id.toString() === id) {
        return node;
      }
      if (node.children.length > 0) {
        const result = this.findNodeById(id, node.children);
        if (result) {
          return result;
        }
      }
    }
    return null;
  }
  // a helper method to move a node to another node
  moveNode(source: TreeNode, target: TreeNode) {
    // remove the source node from its parent
    const sourceParent = this.findParentNode(source);
    if (sourceParent) {
      sourceParent.children = sourceParent.children.filter((child: TreeNode) => child !== source);
    } else {
      this.model = this.model.filter((node: TreeNode) => node !== source);
    }
    // add the source node to the target node's children
    target.children.push(source);
    // expand the target node
    target.expanded = true;
  }

  // a helper method to find the parent node of a node in the model
  findParentNode(node: TreeNode, nodes: TreeNode[] = this.model, parent: TreeNode | null = null): TreeNode | null {
    for (const n of nodes) {
      if (n.children.includes(node)) {
        return n;
      }
      if (n.children.length > 0) {
        const result = this.findParentNode(node, n.children, n);
        if (result) {
          return result;
        }
      }
    }
    return parent;
  }
}

// tree.component.html
<div class="tree">
  <ng-container *ngFor="let node of model">
    <app-tree-node [node]="node" (nodeSelected)="selectNode($event)" (toggleExpanded)="toggleExpanded($event)" (onDragStart)="onDragStart($event, node)" (onDragOver)=
  </ng-container>
</div>

// tree-node.component.ts
import { Component, Input, Output, EventEmitter } from '@angular/core';
import { TreeNode } from './tree-node';

@Component({
  selector: 'app-tree-node',
  templateUrl: './tree-node.component.html',
  styleUrls: ['./tree-node.component.css']
})
export class TreeNodeComponent {
  @Input() node: TreeNode; // the node received from the parent component
  @Output() nodeSelected = new EventEmitter<TreeNode>(); // a custom event to emit when the node is selected
  @Output() toggleExpanded = new EventEmitter<TreeNode>(); // a custom event to emit when the node is expanded or collapsed
  @Output() onDragStart = new EventEmitter<DragEvent>(); // a custom event to emit when the node is dragged
  @Output() onDragOver = new EventEmitter<DragEvent>(); // a custom event to emit when the node is dragged over
  @Output() onDrop = new EventEmitter<DragEvent>(); // a custom event to emit when the node is dropped

  // a method to emit the nodeSelected event
  selectNode() {
    this.nodeSelected.emit(this.node);
  }

  // a method to emit the toggleExpanded event
  toggleNode() {
    this.toggleExpanded.emit(this.node);
  }

  // a method to emit the onDragStart event
  dragStart(event: DragEvent) {
    this.onDragStart.emit(event);
  }

  // a method to emit the onDragOver event
  dragOver(event: DragEvent) {
    this.onDragOver.emit(event);
  }

  // a method to emit the onDrop event
  drop(event: DragEvent) {
    this.onDrop.emit(event);
  }
}

// tree-node.component.html
<div class="tree-node" [class.selected]="node.selected" (click)="selectNode()" draggable="true" (dragstart)="dragStart($event)" (dragover)="dragOver($event)" (drop)="
  <div class="tree-node-content">
    <span class="tree-node-expand-icon" *ngIf="node.children.length > 0" (click)="toggleNode()">
      <i class="fa" [class.fa-plus]="!node.expanded" [class.fa-minus]="node.expanded"></i>
    </span>
    <img class="tree-node-icon" [src]="node.icon" alt="node.icon">
    <span class="tree-node-label">{{ node.label }}</span>
    <span class="tree-node-action-buttons" *ngFor="let button of node.actionButtons">
      <button class="tree-node-action-button" [class]="button.iconClass" [title]="button.tooltip" (click)="button.action(node)"></button>
    </span>
  </div>
  <div class="tree-node-children" *ngIf="node.children.length > 0 && node.expanded">
    <app-tree [model]="node.children" (nodeSelected)="selectNode($event)" (toggleExpanded)="toggleNode($event)" (onDragStart)="dragStart($event)" (onDragOver)="dragOv
  </div>
</div>
```

I hope this code example helps you to implement your Angular TreeNode component. If you want to learn more about Angular and how to work with tree components, you can check out some of the web search results I found for you:

- TreeNode - angular-tree: This is the official documentation of the angular-tree-component library, which provides a simple yet powerful tree component for Angular. It shows how to use the TreeNode class, the TreeModel class, the TreeComponent class, and the custom templates and events.
- Tree | NG-ZORRO: This